#### Inheritance EG. 2: The Point2D and Point3D Classes

Diagram

Description automatically generated

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | /\*  \* A test driver for the Point2D and Point3D classes  \*/  public class TestPoint2DPoint3D {  public static void main(String[] args) {  /\* Test Point2D \*/  // Test constructors and toString()  Point2D p2a = new Point2D(1, 2);  Console.WriteLine(p2a); // toString()  Point2D p2b = new Point2D(); // default constructor  Console.WriteLine(p2b);  // Test Setters and Getters  p2a.X = 3; // Test setters  p2a.Y = 4;  Console.WriteLine(p2a); // toString()  Console.WriteLine("x is: " + p2a.X);  Console.WriteLine("x is: " + p2a.Y);  /\* Test Point3D \*/  // Test constructors and toString()  Point3D p3a = new Point3D(11, 12, 13);  Console.WriteLine(p3a); // toString()  Point2D p3b = new Point3D(); // default constructor  Console.WriteLine(p3b);  // Test Setters and Getters  p3a.X = 21; // in baseclass  p3a.Y = 22; // in baseclass  p3a.Z = 23; // in this class  Console.WriteLine(p3a); // toString()  Console.WriteLine("x is: " + p3a.X); // in baseclass  Console.WriteLine("y is: " + p3a.Y); // in baseclass  Console.WriteLine("z is: " + p3a.Z); // in this class  }  } |

**Inheritance EG. 3: Baseclass Person and its Subclasses**

Diagram

Description automatically generated

/\*

\* A test driver for Person and its subclasses.

\*/

public class **TestPerson** {

public static void main(String[] args) {

/\* Test Student class \*/

Student s1 = new Student("Tan Ah Teck", "1 Happy Ave");

s1.AddCourseGrade("IM101", 97);

s1.AddCourseGrade("IM102", 68);

s1.PrintGrades();

Console.WriteLine("Average is " + s1.GetAverageGrade());

/\* Test Teacher class \*/

Teacher t1 = new Teacher("Paul Tan", "8 sunset way");

Console.WriteLine(t1);

String[] courses = {"IM101", "IM102", "IM101"};

for (String course: courses) {

if (t1.AddCourse(course)) {

Console.WriteLine(course + " added.");

} else {

Console.WriteLine(course + " cannot be added.");

}

}

for (String course: courses) {

if (t1.RemoveCourse(course)) {

Console.WriteLine(course + " removed.");

} else {

Console.WriteLine(course + " cannot be removed.");

}

}

}

}

Tan Ah Teck(1 Happy Ave)

Tan Ah Teck

8 Sunrise Place

Student: Mohd Ali(8 Kg Java)

Mohd Ali

9 Kg Satu

Student: Mohd Ali(9 Kg Satu) IM101:97 IM102:68

Average is: 82.5

Teacher: Paul Tan(8 sunset way)

IM101 added.

IM102 added.

IM101 cannot be added.

IM101 removed.

IM102 removed.

IM101 cannot be removed.

#### Ex: Baseclass Shape and its subclasses Circle, Rectangle and Square

Diagram

Description automatically generated

a) Write a baseclass called Shape (as shown in the class diagram), which contains:

* Two properties Color (String) and Filled (boolean).
* Two constructors: a no-arg (no-argument) constructor that initializes the Color to "green" and Filled to true, and a constructor that initializes the color and filled to the given values.
* A ToString() method that returns "A Shape with color of xxx and filled/Not filled".

**Write a test program to test all the methods defined in Shape.**

b) Write two subclasses of Shape called Circle and Rectangle, as shown in the class diagram.

The Circle class contains:

* A property Radius (double).
* Three constructors as shown. The no-arg constructor initializes the Radius to 1.0.
* Methods GetArea() and GetPerimeter().
* Override the ToString() method inherited, to return "A Circle with radius=xxx, which is a subclass of yyy", where yyy is the output of the toString() method from the baseclass.

The Rectangle class contains:

* Two properties Width (double) and Length (double).
* Three constructors as shown. The no-arg constructor initializes the Width and Length to 1.0.
* Methods GetArea() and GetPerimeter().
* Override the ToString() method inherited, to return "A Rectangle with width=xxx and length=zzz, which is a subclass of yyy", where yyy is the output of the toString()method from the baseclass.

**Write a test program to test all the methods defined in Circle and Rectangle.**

c) Write a class called Square, as a subclass of Rectangle. Convince yourself that Square can be modeled as a subclass of Rectangle.

*Square inherites and overrides 2 properties Width and Length from base class in such a way that Width and Length are always equal.*

*Square has Side property which does not have a private variable. Getter and setter of Side property should return and update Length or Width in such a way that Width and Length are always equal.*

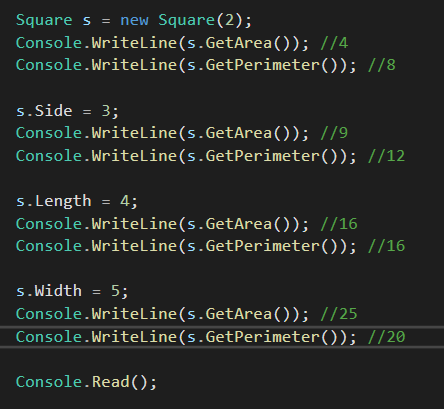
* Provide the appropriate constructors (as shown in the class diagram). Hint:
* public Square(double side) : base(side, side) // Call baseclass Rectangle(double, double)

{

}

* Override the ToString() method to return "A Square with side=xxx, which is a subclass of yyy", where yyy is the output of the ToString() method from the baseclass.
* Do you need to override the getArea() and getPerimeter()? Try them out.

**Write a test program to test all the methods defined in Square.**

****